Program slicing: a survey

Härmel Nestra

Institute of Computer Science

University of Tartu

e-mail: *nestra@math.ut.ee*

Introduction

Definition

Informal Definition

- A **slicing criterion** is a list of pairs of program points and memory locations (e.g. variables).
  - Or equivalently, it is a function from program points to memory location sets.
- A **slice** of a program P w.r.t a criterion γ is a subset of P consisting of precisisely those statements which are relevant to γ.
- **Program slicing** is an action with the aim of finding slices.

Three-dimensional classification

- Executable or not?
  - In **executable** slicing, a subset of a program is required to be executable.
  - If we are not speaking of executable slicing, finding a subset means just giving a rule saying which elementary code units (whatever they are...) are thrown out.
- Backward or forward?
  - In **backward** slicing, we are interested in the statements of the program which can influence the values at points of the criterion.
  - In **forward** slicing, we are interested in the statements of the program which can be influenced by the values at points of the criterion.
- Static or dynamic?
  - **Static** slicing is performed using no run-time information.
  - **Dynamic** slicing uses information about user inputs etc.
    - Using run-time information keeps the slices smaller.

Kinds of slicing

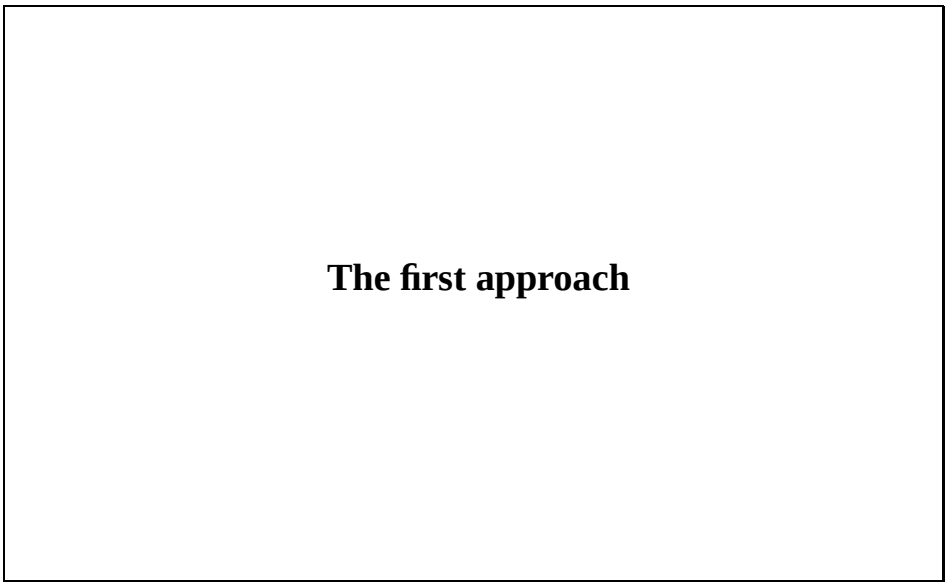
- So we have 8 different kinds of slicing.
- **Executable backward static** slicing occurred first in research history.

Motivation

Applications of slicing

- Parallelizing a sequential program.
- Debugging.
  - Slicing some parts away helps us to localize bugs in a large program.
  - Finding the forward slice of an erroneous command can give ideas how to correct the program.
  - Finding dead code (probably come into being due to a bug).
- Testing, maintenance.
  - Only parts of the software affected by new modifi cations have to be tested.
- …

More closely on backward static slicing


The first approach

More rigorous specification

A subset Q of a program P is a slice of P w.r.t. criterion γ if, for any initial state, programs P and Q compute the same values at the points of γ.

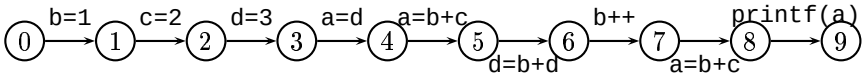
Example program

Consider the following toy program:

```
b = 1;
c = 2;
d = 3;
a = d;
a = b + c;
d = b + d;
b++;
a = b + c;
printf(a);
```

A slicing criterion

We can take the following to be its control flow graph:



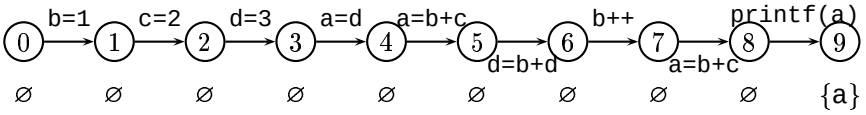
Consider the slicing criterion

$$\{(9, \mathsf{a})\}\,.$$

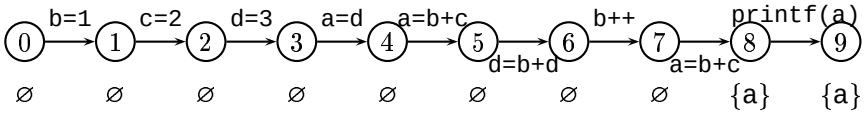
The slice

```
b = 1;          ⇒    b = 1;
c = 2;               c = 2;
d = 3;
a = d;
a = b + c;
d = b + d;
b++;                 b++;
a = b + c;           a = b + c;
printf(a);
```

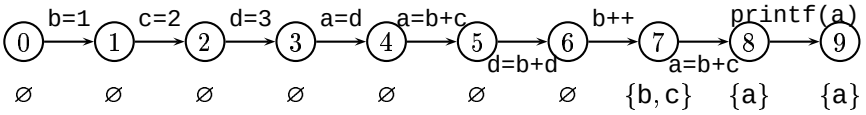
Relevant Sets analysis 1



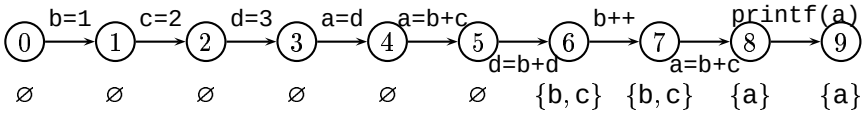
Relevant Sets analysis 2



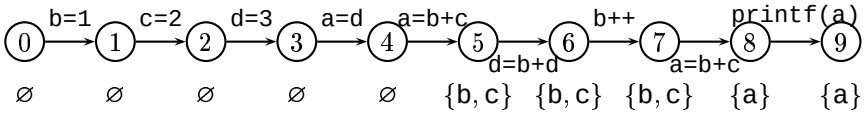
Relevant Sets analysis 3



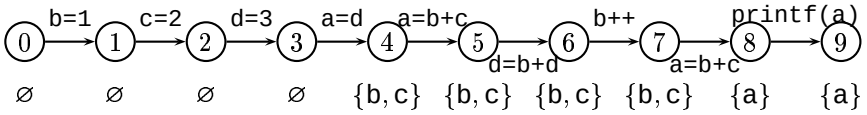
Relevant Sets analysis 4



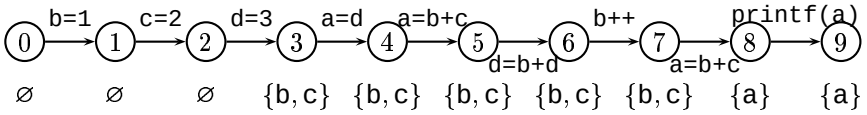
Relevant Sets analysis 5



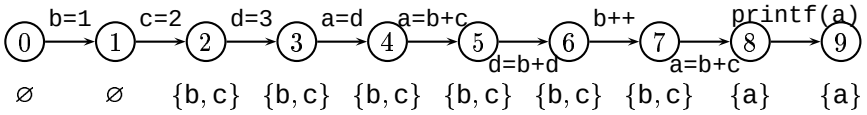
Relevant Sets analysis 6



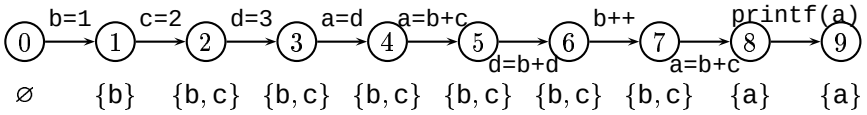
Relevant Sets analysis 7



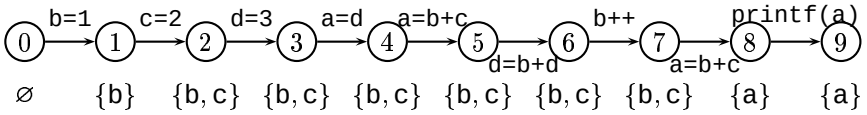
Relevant Sets analysis 8



Relevant Sets analysis 9



Relevant Sets analysis 10



Obtaining the slice: the first approximation

- Take the set of edges where a location relevant at its end vertex is updated. The desired slice corresponds to this set.

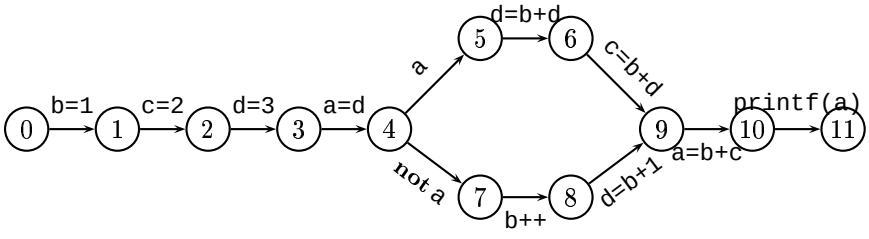
Next example

Consider the following program:

```
b = 1;
c = 2;
d = 3;
a = d;
if (a) {
  d = b + d;
  c = b + d;
} else {
  b++;
  d = b + 1;
}
a = b + c;
printf(a);
```

Specifying the task

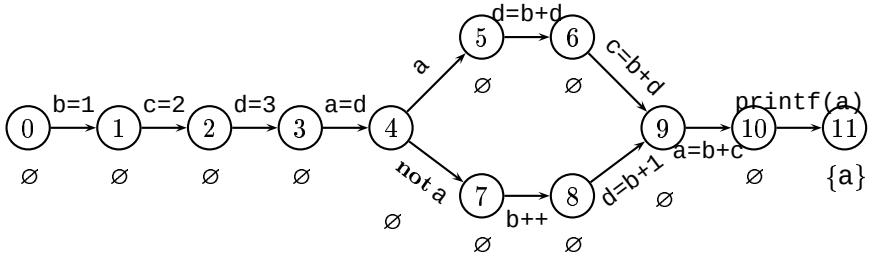
Take the control flow graph as follows:



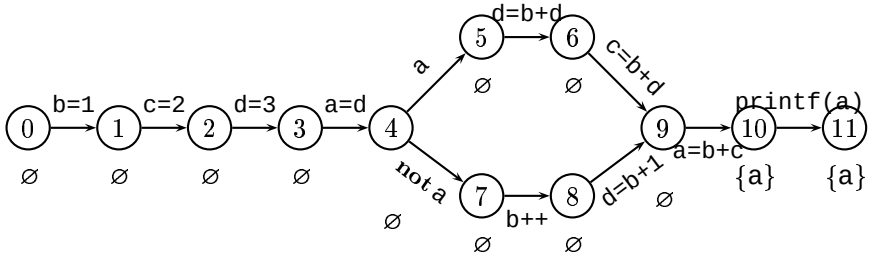
Consider the slicing criterion

$$\{(11, \mathsf{a})\}\,.$$

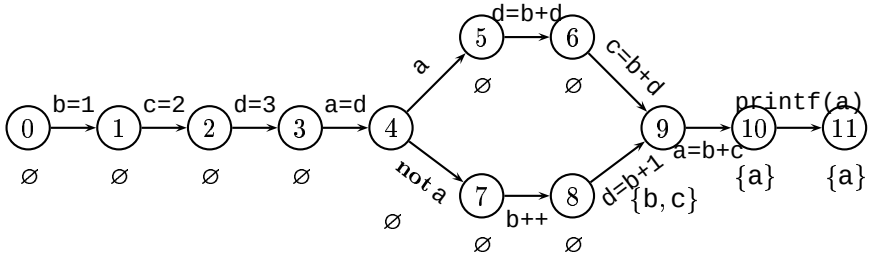
Relevant Sets analysis 1



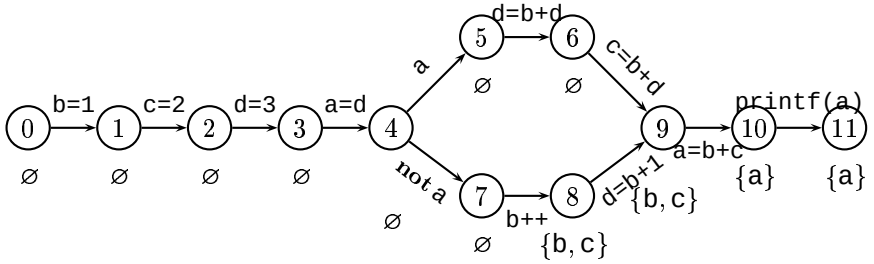
Relevant Sets analysis 2



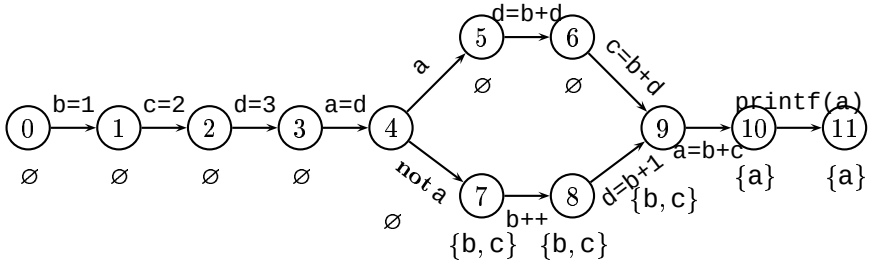
Relevant Sets analysis 3



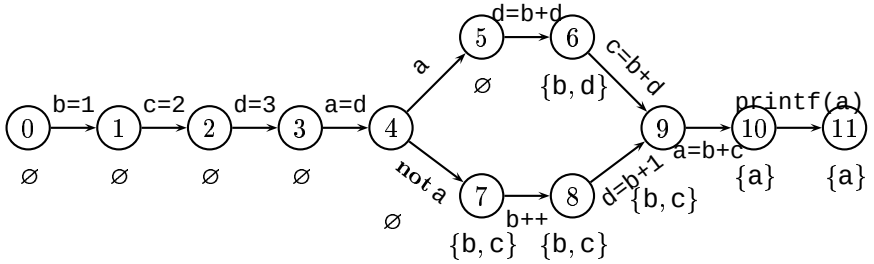
Relevant Sets analysis 4



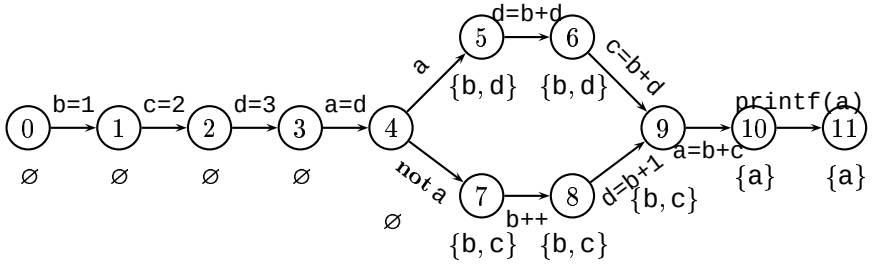
Relevant Sets analysis 5



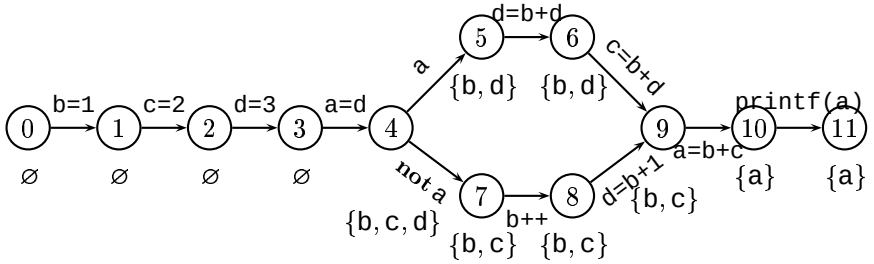
Relevant Sets analysis 6



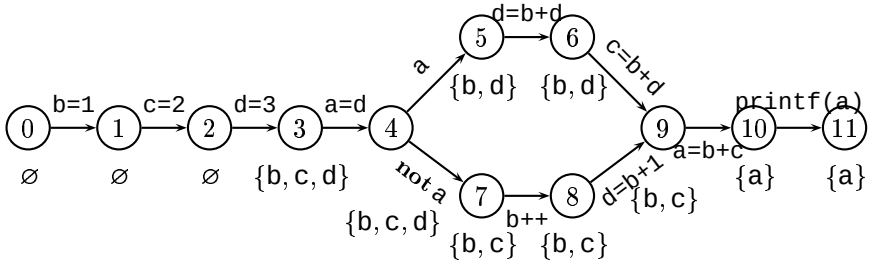
Relevant Sets analysis 7



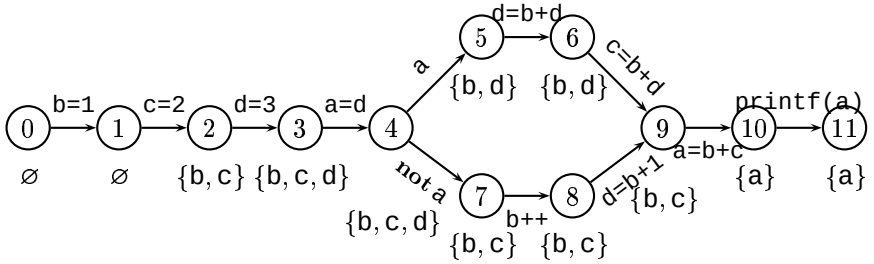
Relevant Sets analysis 8



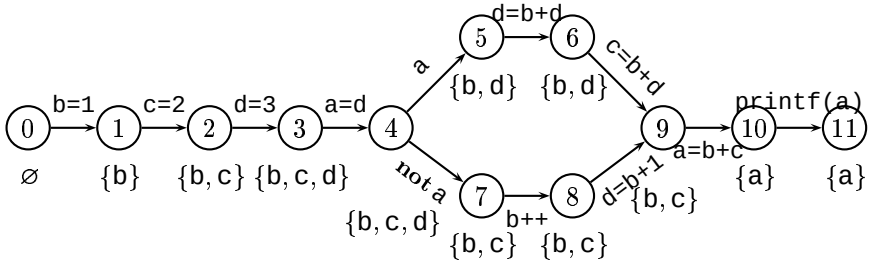
Relevant Sets analysis 9



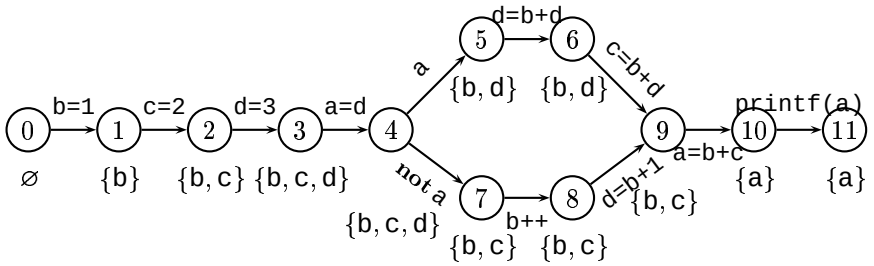
Relevant Sets analysis 10



Relevant Sets analysis 11



Relevant Sets analysis 12



Slice?

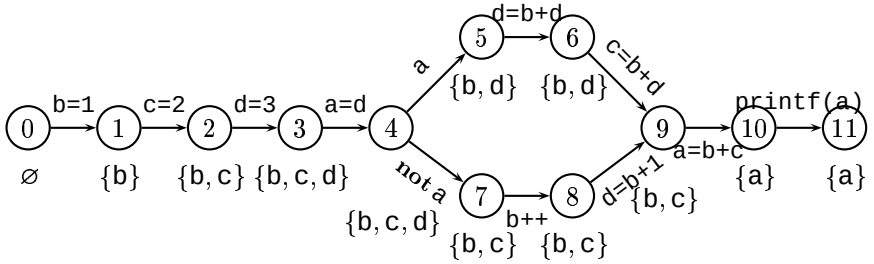
- If a control statement contains a line of the slice, it is also taken into the slice.

```
b = 1;                    ?      b = 1;
c = 2;                    ⇒      c = 2;
d = 3;                           d = 3;
a = d;
if (a) {                         if (a) {
  d = b + d;                       d = b + d;
  c = b + d;                       c = b + d;
} else {                         } else {
  b++;                             b++;
  d = b + 1;
}                                }
a = b + c;                       a = b + c;
printf(a);
```

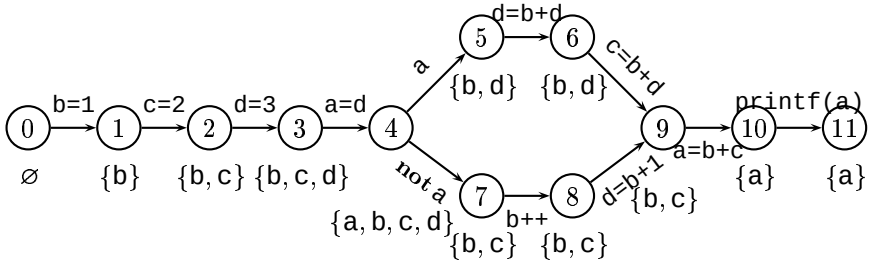
Control statements are specific

If a control statement is in the slice, all the variables of its test expression must be declared relevant at that point!

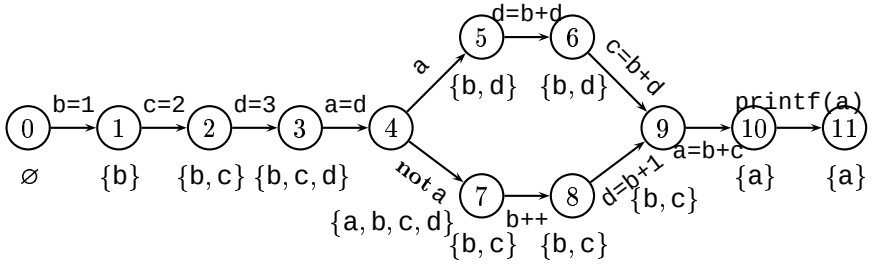
Relevant Sets analysis continued 1



Relevant Sets analysis continued 2



Relevant Sets analysis continued 3



The right slice

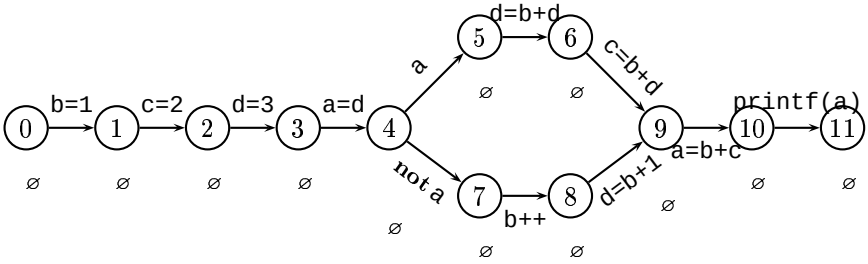
```
b = 1;              ⇒     b = 1;
c = 2;                    c = 2;
d = 3;                    d = 3;
a = d;                    a = d;
if (a) {                  if (a) {
  d = b + d;                d = b + d;
  c = b + d;                c = b + d;
} else {                  } else {
  b++;                      b++;
  d = b + 1;
}                         }
a = b + c;                a = b + c;
printf(a);
```

Control dependence

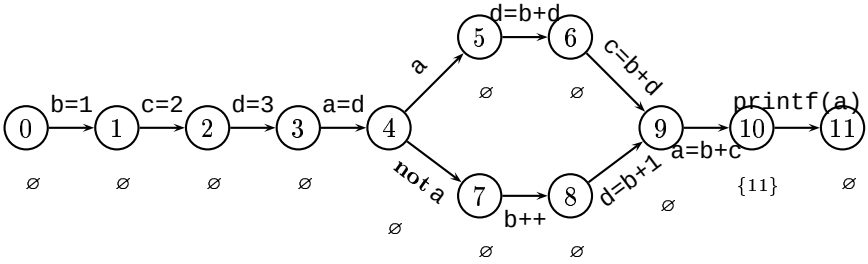
Let G be a directed graph with marked end-vertices and u, v any vertices.

- Call v **post-dominating** u iff any path from u to any end-vertex uses v.
- Call v **control dependent** on u iff both of the following hold:
  a. v does not post-dominate u;
  b. there exists a successor w of u such that v post-dominates w.

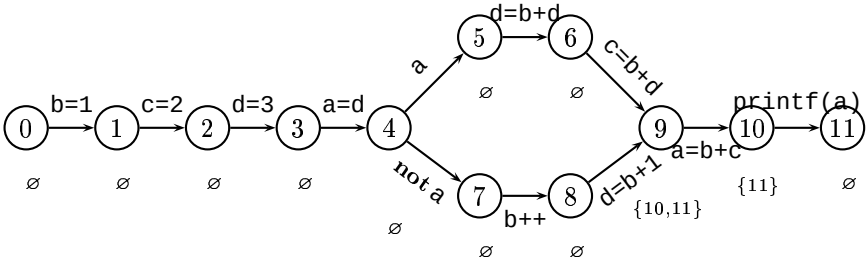
Post-dominance analysis 1



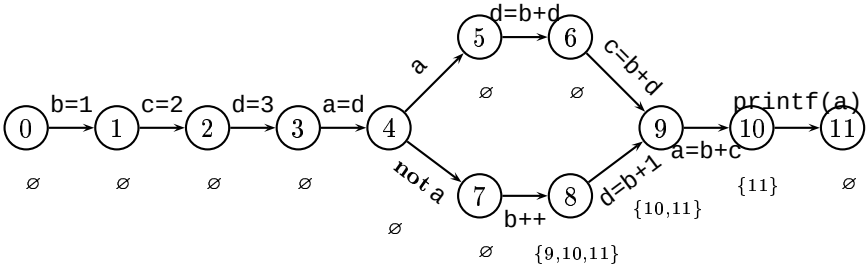
Post-dominance analysis 2



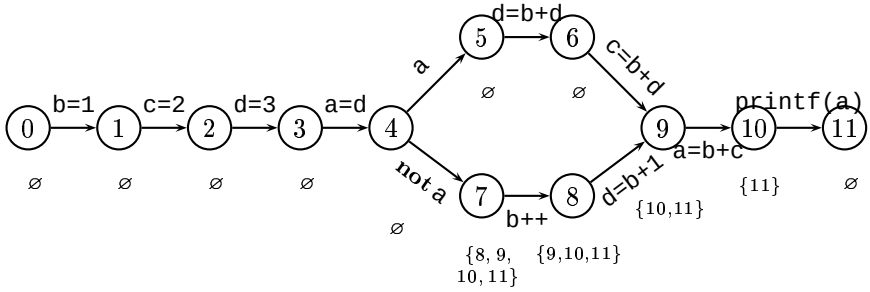
Post-dominance analysis 3



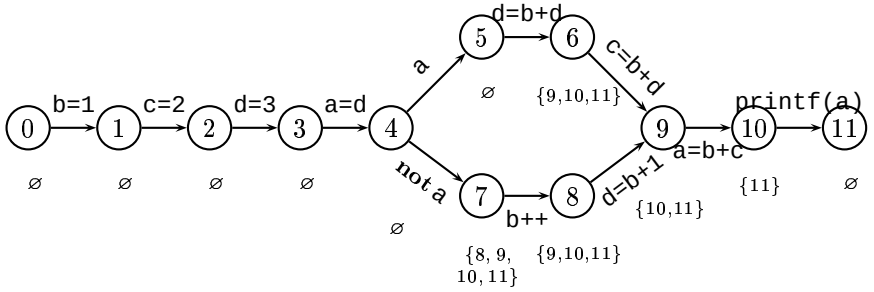
Post-dominance analysis 4



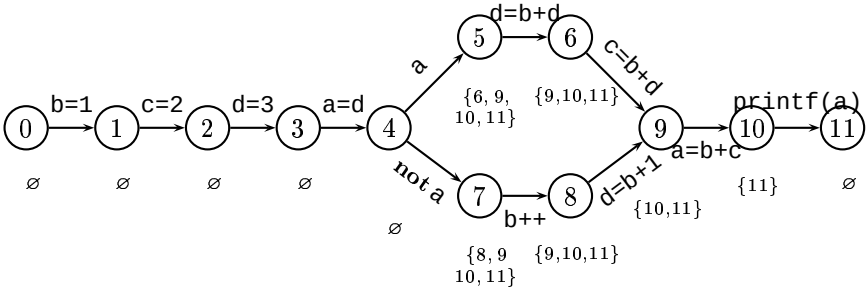
Post-dominance analysis 5



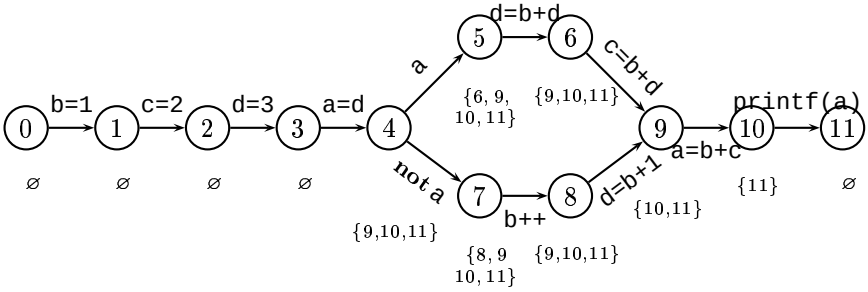
Post-dominance analysis 6



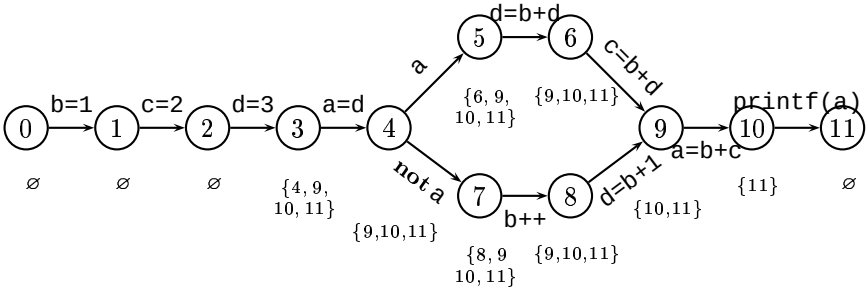
Post-dominance analysis 7



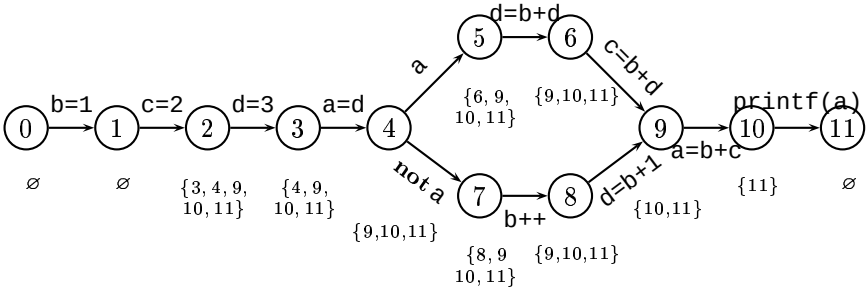
Post-dominance analysis 8



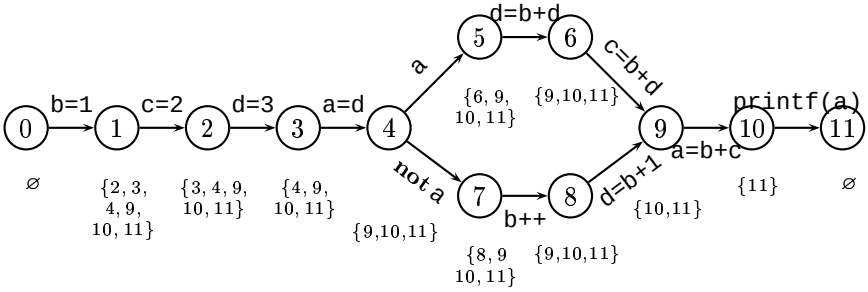
Post-dominance analysis 9



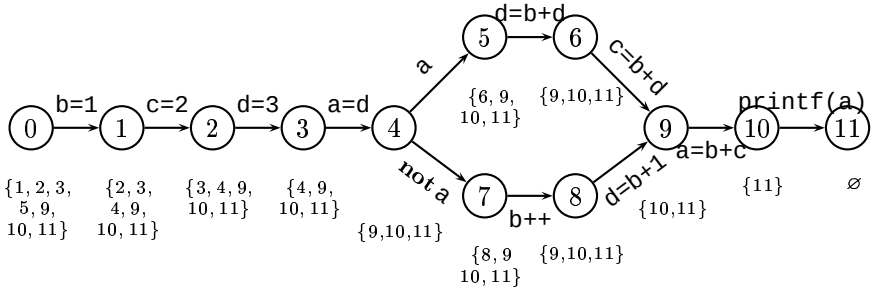
Post-dominance analysis 10



Post-dominance analysis 11



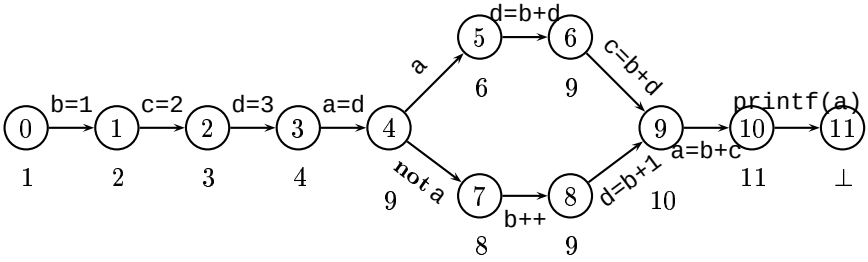
Post-dominance analysis 12



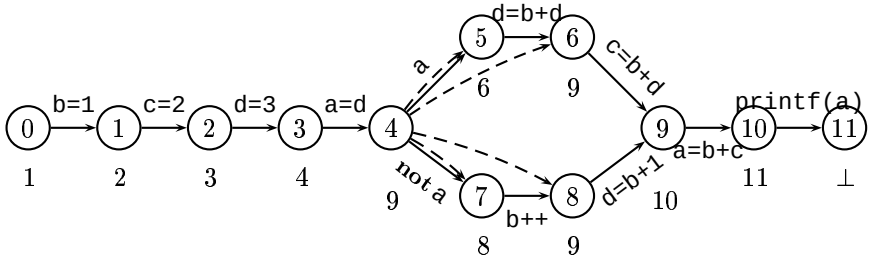
Immediate post-dominators

- The post-dominance relation is an order.
- The least w.r.t. the post-dominance order element among the strict post-dominators of u is called the **immediate post-dominator** of u.
- Every vertex except the end vertices has the immediate post-dominator.

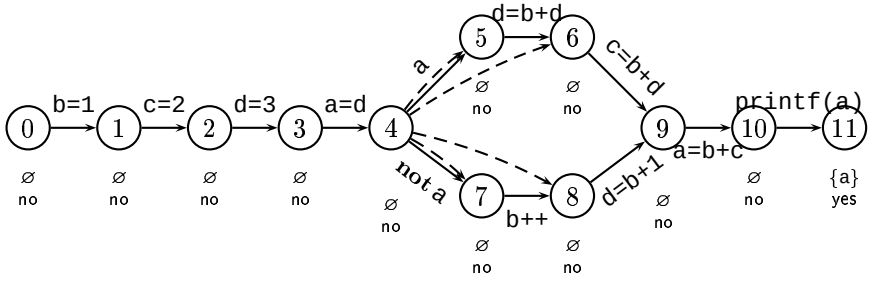
Immediate post-dominators and control dependence 1



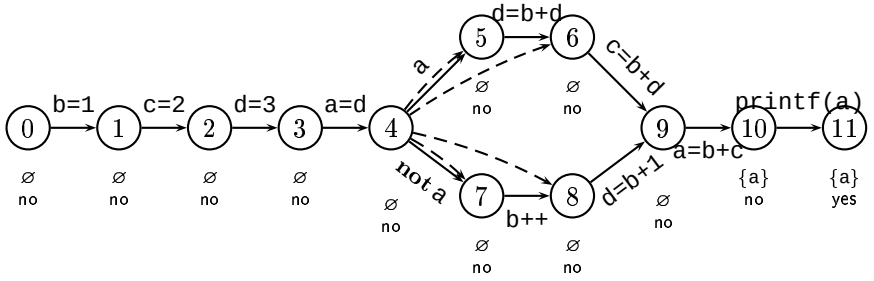
Immediate post-dominators and control dependence 2



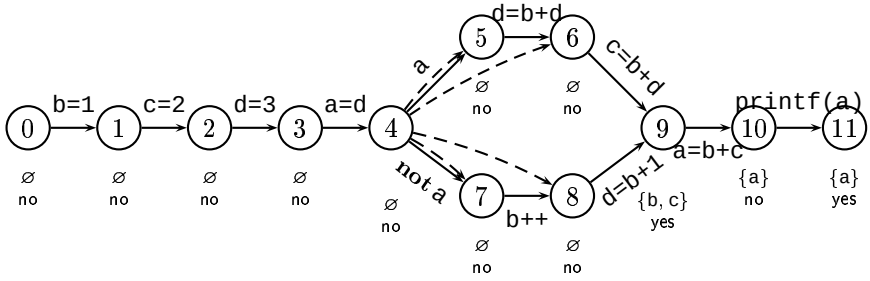
Relevant Sets analysis on CFG + CD 1



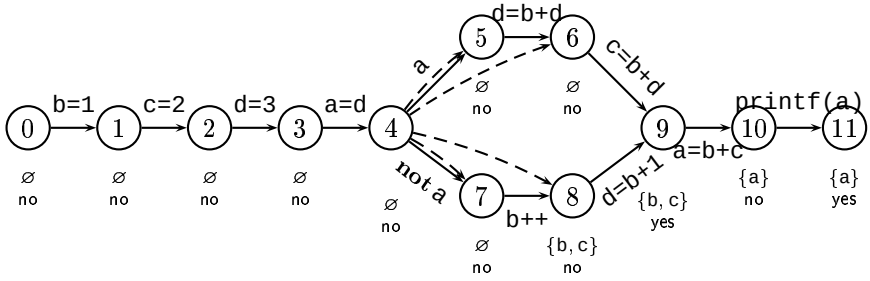
Relevant Sets analysis on CFG + CD 2



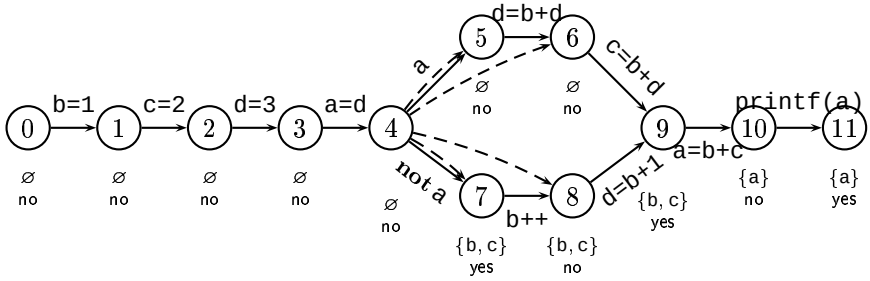
Relevant Sets analysis on CFG + CD 3



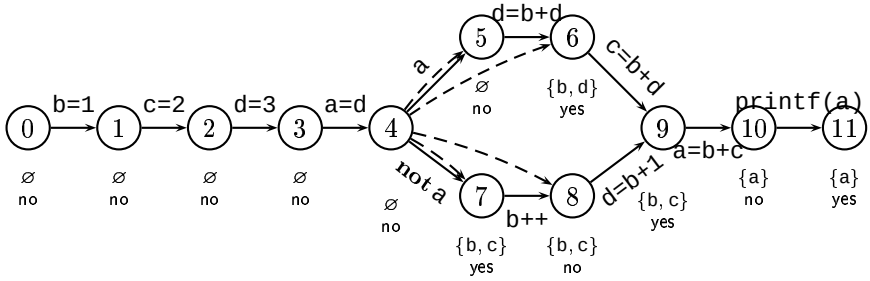
Relevant Sets analysis on CFG + CD 4



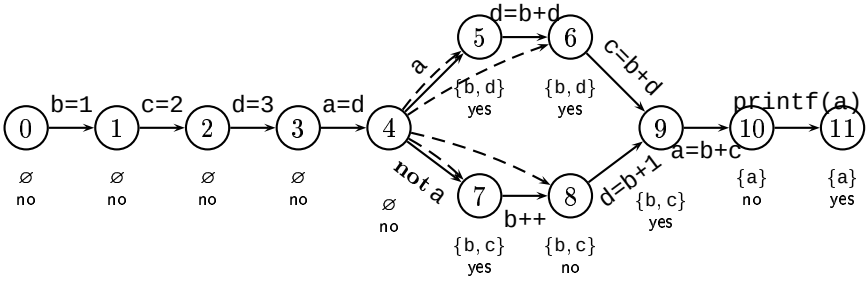
Relevant Sets analysis on CFG + CD 5



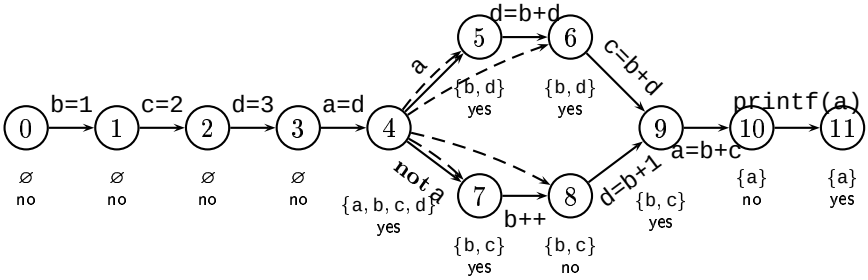
Relevant Sets analysis on CFG + CD 6



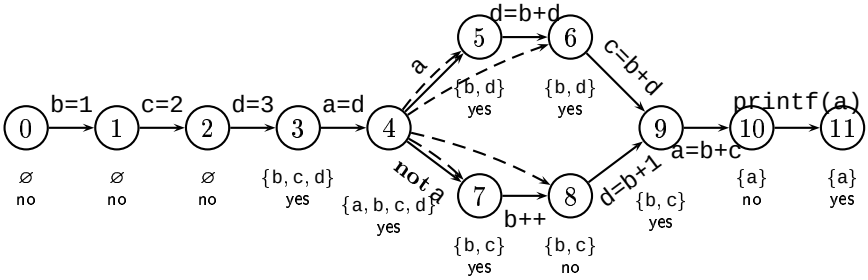
Relevant Sets analysis on CFG + CD 7



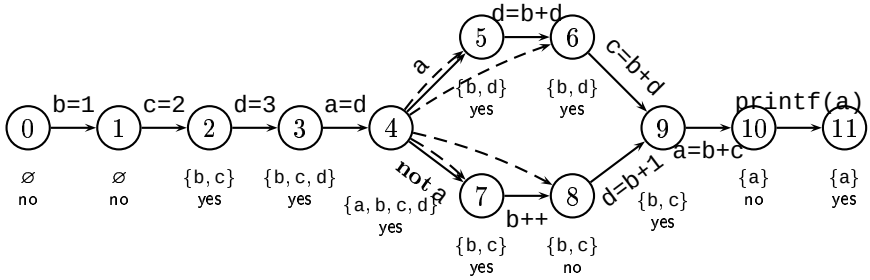
Relevant Sets analysis on CFG + CD 8



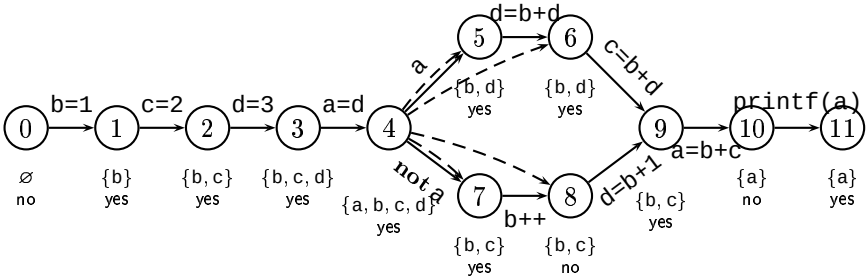
Relevant Sets analysis on CFG + CD 9



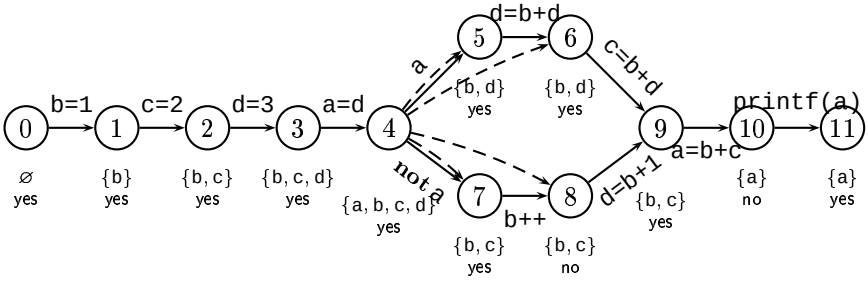
Relevant Sets analysis on CFG + CD 10

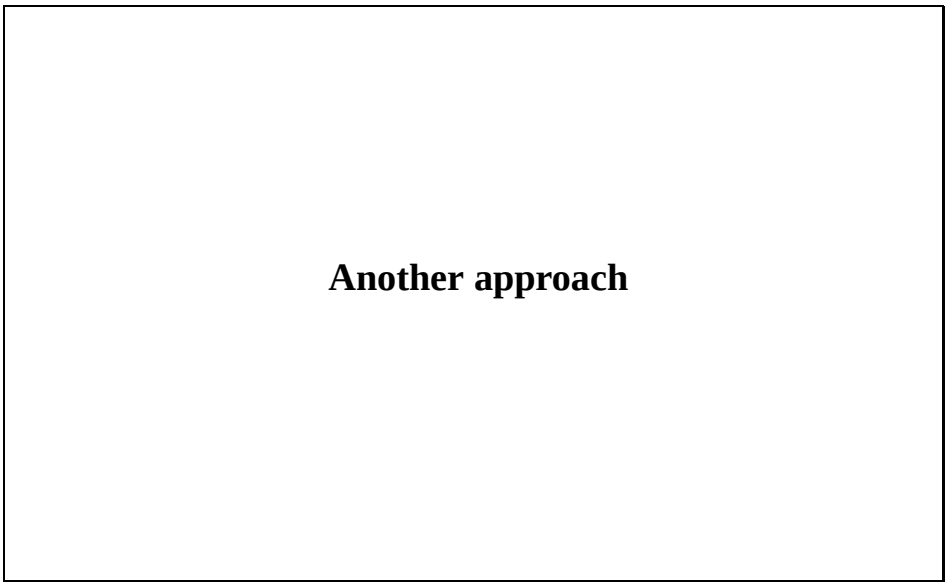


Relevant Sets analysis on CFG + CD 11



Relevant Sets analysis on CFG + CD 12




Another approach

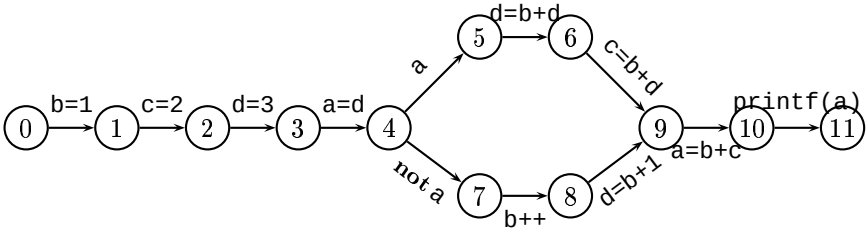
New concepts

- **Reaching Definitions** analysis computes for every program point, at which program points the initialized variables can be last updated.
- A vertex v of control flow graph is said to be **data dependent** on a vertex u iff v can read a location which can be last updated at u.

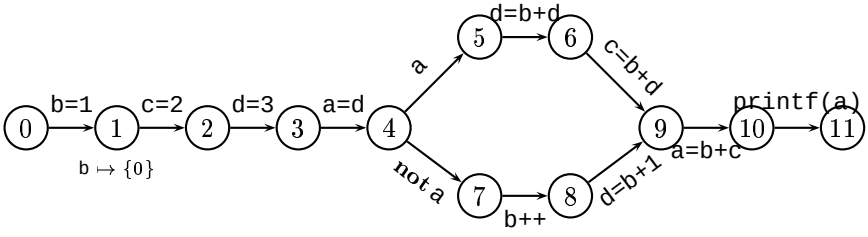
Plan

- Perform Reaching Definitions.
- Compute control dependences.
- Compute data dependences of the program.
- Compute criterion-specific data dependences.
  - Any pair $(p, x) \in \gamma$ is treated as using variable x at p. This generally adds some new dependences.
- The slice can be obtained as the set of vertices reachable from points mentioned by the criterion in the graph whose edges are the reversed data and control dependence ones.

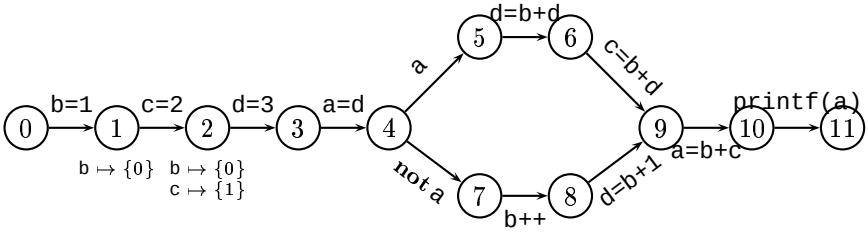
Reaching Definitions analysis on the last example 1



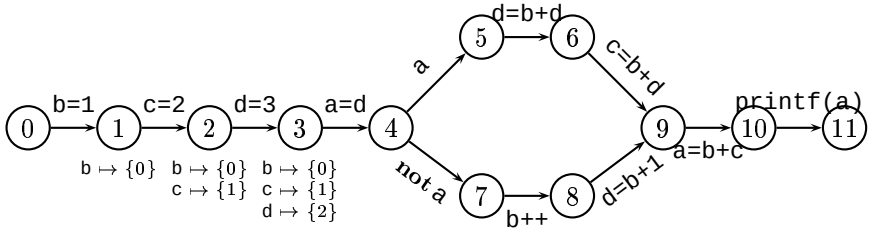
Reaching Definitions analysis on the last example 2



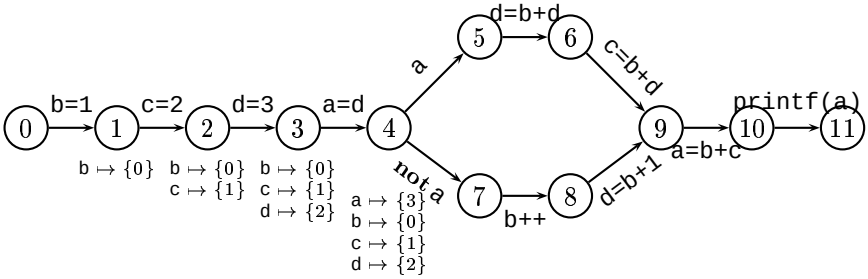
Reaching Definitions analysis on the last example 3



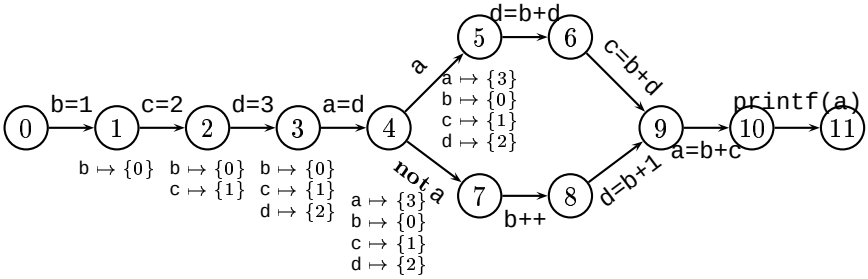
Reaching Definitions analysis on the last example 4



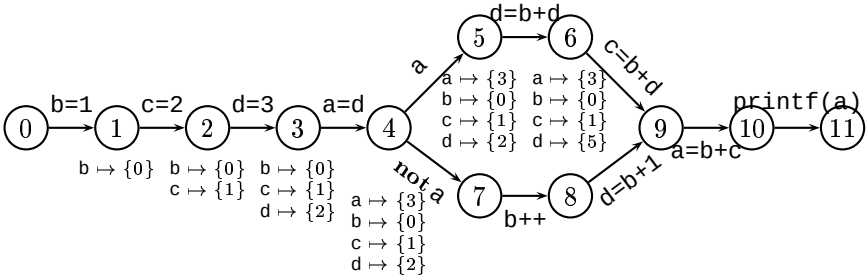
Reaching Definitions analysis on the last example 5



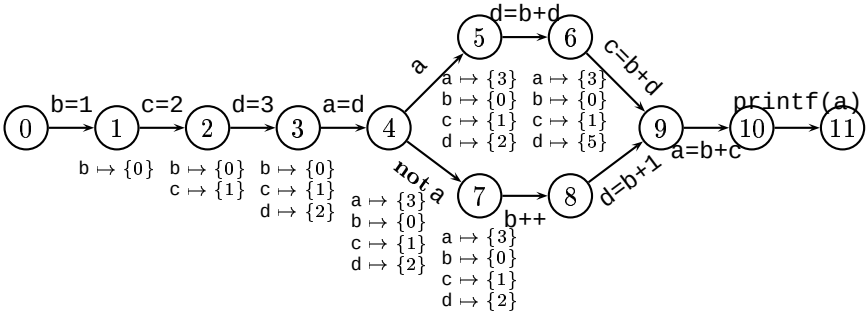
Reaching Definitions analysis on the last example 6



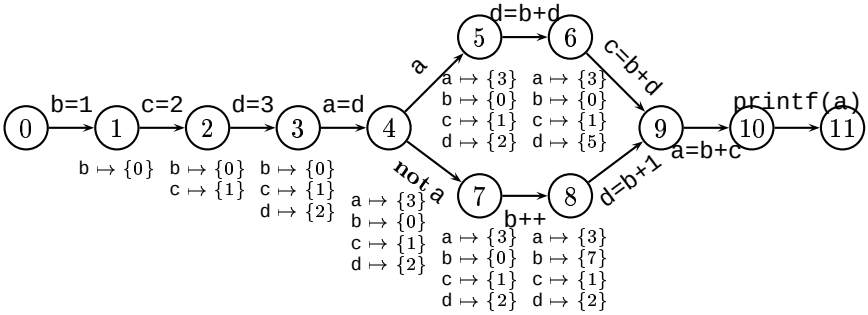
Reaching Definitions analysis on the last example 7



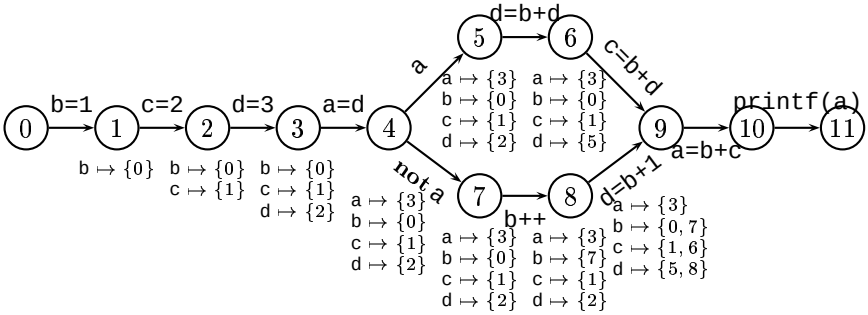
Reaching Definitions analysis on the last example 8



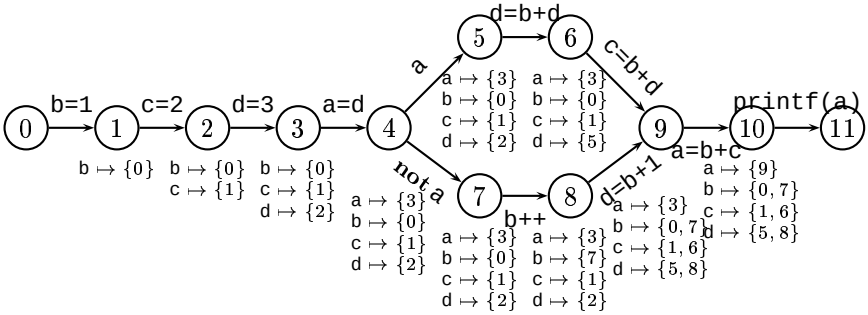
Reaching Definitions analysis on the last example 9



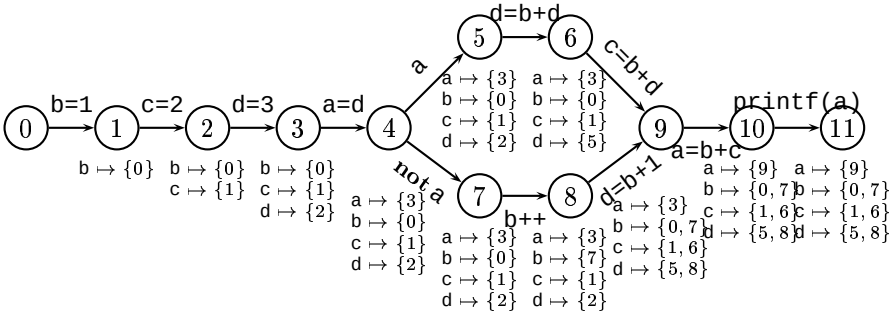
Reaching Definitions analysis on the last example 10



Reaching Definitions analysis on the last example 11

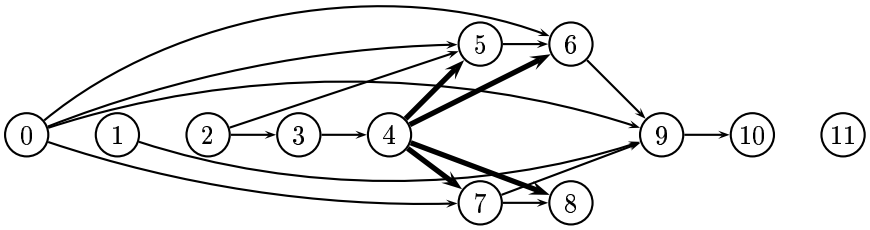


Reaching Definitions analysis on the last example 12



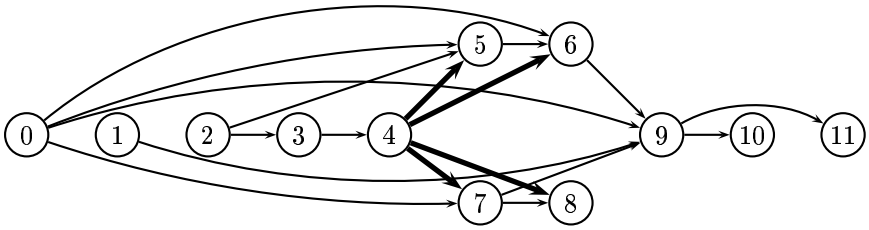
Data and control dependences

Thin arrows denote data dependences, bold arrows denote control dependences.



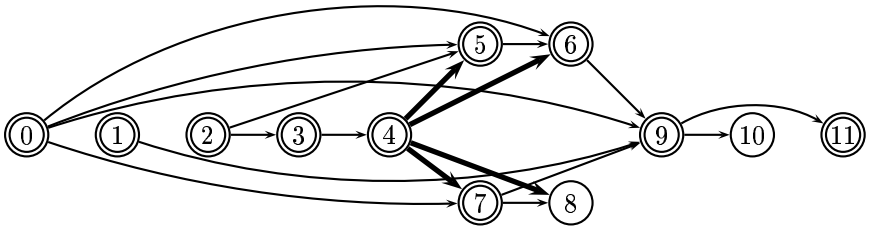
Data and control dependences

Thin arrows denote data dependences, bold arrows denote control dependences.



Data and control dependences

Thin arrows denote data dependences, bold arrows denote control dependences.



Conclusion

Comparision

- Slicing via Relevant Sets analysis depends wholly on the criterion.
- Reaching Definitions does not depend on the criterion. Using the second approach, only the last cheap steps use the criterion.
  - So if one has to slice a program w.r.t. many criterions, the second approach is better!

Further

- This was intraprocedural slicing only…